Boolean Algebra
&
Logic Circuits

Dr. Ahmed El-Bialy
Dr. Sahar Fawzy

Binary Logic

- Binary logic deals with binary variables that take on two discrete values and with the operations of mathematical logic applied to these variables.

	Regular Algebra	Boolean Algebra
Values	Numbers Integers Real Numbers Complex Numbers	Zero (0) One (1)
Operators	Add, Subtract Multiply, Divide Logarithm, etc.	AND OR Complement

Logic Gates

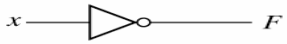
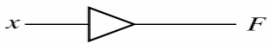
Name	Graphic symbol	Algebraic function	Truth table
AND		$F = xy$	x y | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$F = x + y$	x y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
Inverter		$F = x'$	x | F 0 | 1 1 | 0
Buffer		$F = x$	x | F 0 | 0 1 | 1
NAND		$F = (xy)'$	x y | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = (x + y)'$	x y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
Exclusive-OR (XOR)		$F = xy' + x'y$ $= x \oplus y$	x y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
Exclusive-NOR or equivalence		$F = xy + x'y'$ $= (x \oplus y)'$	x y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

Fig. 2-5 Digital logic gates

Truth Table

- A truth table is a table of combinations of the binary variables showing the relationship between the values that the variables take on and the values of the result of the operation
- If there are n inputs, there will be 2^n rows in the table

A	B	C = A • B
0	0	0
0	1	0
1	0	0
1	1	1

AND Operation

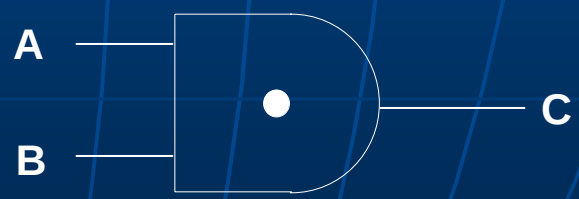
- Represented by a dot (.) or by the absence of an operator
- The definition of logical (.) is:

$$0 \cdot 0 = 0 \qquad 0 \cdot 1 = 0$$

$$1 \cdot 0 = 0 \qquad 1 \cdot 1 = 1$$

- It looks like multiplication, but it is not. Therefore symbol is used for AND instead of a dot.

A	B	C = A • B
0	0	0
0	1	0
1	0	0
1	1	1



OR Operation

- Represented by a plus (+)
- The definition of logical (+) is:

$$0 + 0 = 0 \qquad 0 + 1 = 1$$

$$1 + 0 = 1 \qquad 1 + 1 = 1$$

- It looks like addition, but it is not. Therefore symbol is used for OR instead of a plus.

A	B	C = A + B
0	0	0
0	1	1
1	0	1
1	1	1

Not (Complement) Operation

- Represented by a bar over the variable or a prime (’)
- The definition of logical (’) is:

0’ = 1

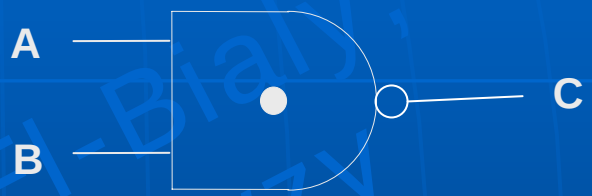
1’ = 0

If X = 1 then X’ = 0

If X = 0 then X’ = 1

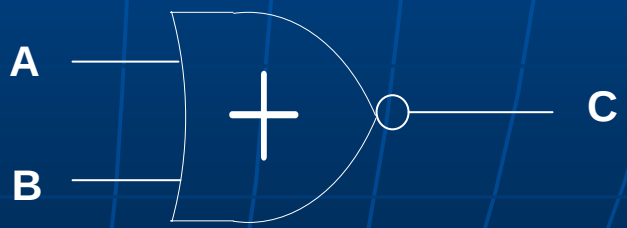
- NAND: the complement of the AND operation

A	B	C =(A • B)’
0	0	1
0	1	1
1	0	1
1	1	0



- NOR: the complement of the OR operation

A	B	C = (A + B)'
0	0	1
0	1	0
1	0	0
1	1	0



- XOR: exclusive OR is represented by $\oplus$

  Its definition logic is:

  $0 \oplus 0 = 0$ $\quad$ $0 \oplus 1 = 1$

  $1 \oplus 0 = 1$ $\quad$ $1 \oplus 1 = 0$

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

- XNOR: exclusive-NOR is the complement of the exclusive-OR and expressed as:

  $(A \oplus B)' = AB + A'B'$

- Exclusive-OR Theorem:

$X \oplus 0 = X$ $\quad\quad$ $X \oplus 1 = X'$

$X \oplus X = 0$ $\quad\quad$ $X \oplus X' = 1$

$X \oplus Y = Y \oplus X$ $\quad\quad$ $(X \oplus Y)' = X \oplus Y' = X' \oplus Y$

$(X \oplus Y) \oplus Z = X \oplus (Y \oplus Z) = X \oplus Y \oplus Z$

- Odd function: multiple-variable exclusive-OR operation : is one whenever the corresponding binary truth-table values have an odd number of 1’s.

Boolean Expression

- Boolean expressions are made up of variables combined by Logical operations .
- Examples: **[A B (C + B’) + D]** **B • E** **C’**
- **Literals** each instance of a variable

This expression has 4 variables and 10 literals:

a’bd + bcd + ac’ + a’d’

Boolean Expression

- Boolean expressions can be represented in a truth table which lists all possible combinations of the values of all variables in the expression.

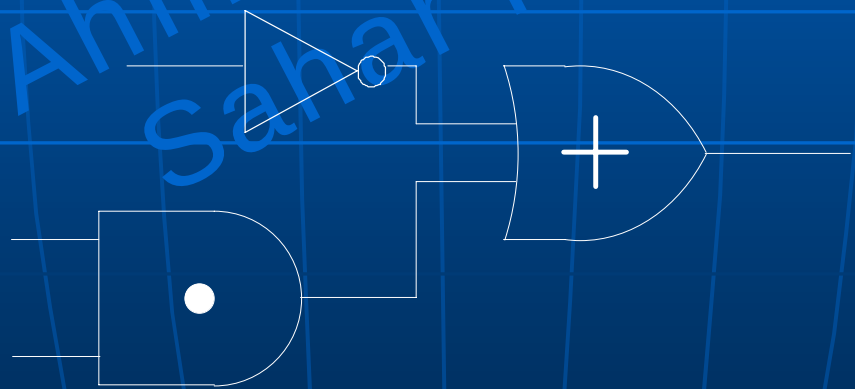
$$F = A' + B\,C$$

A	B	C	A'	B C	F = A' + B C
0	0	0	1	0	1
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	0	1	1

Boolean Expression

- Boolean expressions can be transformed from an algebraic expression into a circuit diagram composed of logic gates.

$$F = A' + B\,C$$



Basic Boolean Algebra Theorems

$X+0=X$	$X \cdot 1=X$
$X+1=1$	$X \cdot 0=0$
$X+X=X$	$X \cdot X=X$
$X+X'=1$	$X \cdot X'=0$

$$(X')'=X$$

Commutative Law:

$$X \cdot Y = Y \cdot X \qquad X + Y = Y + X$$

Associative Law:

$$(X + Y) + Z = X + (Y + Z) = X + Y + Z$$

$$(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z) = X \cdot Y \cdot Z$$

Distributive Law: $X (Y + Z) = X Y + X Z$

Basic Boolean Algebra Theorems

Other Distributive Law:

$$X + Y Z = (X + Y) (X + Z)$$

Proof:

$$\begin{aligned}
(X + Y) (X + Z) &= X (X + Z) + Y (X + Z) && \text{2-1}\\
&= X X + X Z + Y X + Y Z && \text{2-2}\\
&= X + X Z + X Y + Y Z && \text{2-3}\\
&= X \cdot 1 + X Z + X Y + Y Z && \text{2-4}\\
&= X (1 + Z + Y) + Y Z && \text{2-5}\\
&= X \cdot 1 + Y Z && \text{2-6}\\
&= X + Y Z && \text{2-7}
\end{aligned}$$

Simplification Theorems

$X Y + X Y' = X$ $\qquad$ $(X + Y) (X + Y') = X$

$X + X Y = X$ $\qquad$ $X (X + Y) = X$

$(X + Y') Y = X Y \; X Y' + Y = X + Y$

Basic Boolean Algebra Theorems

DeMogan's Law:

$(X + Y)' = X' Y'$ $\qquad\qquad$ $(X Y)' = X' + Y'$

Proof:

X	Y	X'	Y'	X + Y	(X + Y)'	X'Y'	X Y	(XY)'	X' + Y'
0	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	0	0	1	1
1	0	0	1	1	0	0	0	1	1
1	1	0	0	1	0	0	1	0	0

DeMogan's Law- one step rule:

$[f (X_1, X_2, \ldots X_N, 0, 1, +, \bullet)]' = f (X_1', X_2', \ldots X_N', 1, 0, \bullet, +)$

- Replace all variables with the inverse
- Replace + with • and • with +
- Replace 0 with 1 and 1 with 0
- Be careful of hierarchy ( )

Methods of Boolean Expression Simplification

- Combine terms
- Eliminate terms
- Eliminate literals
- Add redundant terms if need

A’B + A’B’C’D’ + ABCD’

A’(B + B’C’D’) + ABCD’

A’(B + C’D’) + ABCD’

A’B + A’C’D’ + ABCD’

A’C’D’ + B(A’ +ACD’)

A’C’D’ + B(A’ +CD’)

A’B + BCD’ + A’C’D’

Standard Forms

- Product term (e.g., XYZ) (Minterms)
- Sum term (e.g., X+Y+Z) (Maxterms)

They do not imply arithmetic operations in Boolean algebra;

instead, they specify the logical operation AND and OR, respectively

X	Y	Z	Min	Minterms	Max	Maxterms
0	0	0	m0	X'Y'Z'	M0	X+Y+Z
0	0	1	m1	X'Y'Z	M1	X+Y+Z'
0	1	0	m2	X'YZ'	M2	X+Y'+Z
0	1	1	m3	X'YZ	M3	X+Y'+Z'
1	0	0	m4	XY'Z'	M4	X'+Y+Z
1	0	1	m5	XY'Z	M5	X'+Y+Z'
1	1	0	m6	XYZ'	M6	X'+Y'+Z
1	1	1	m7	XYZ	M7	X'+Y'+Z'

Standard Forms

Example: A Boolean function F is equal to 1 for each of the following binary combinations of the variables X, Y, and Z: 000, 001,100, 110.

Derive its algebraic expression

→ Truth table and logic sum of minterms

X	Y	Z	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Sum-of-Product Boolean expressions for F:

$S = m_0+m_1+m_4+m_6$

$=X'Y'Z'+X'Y'Z+XY'Z'+XYZ'$

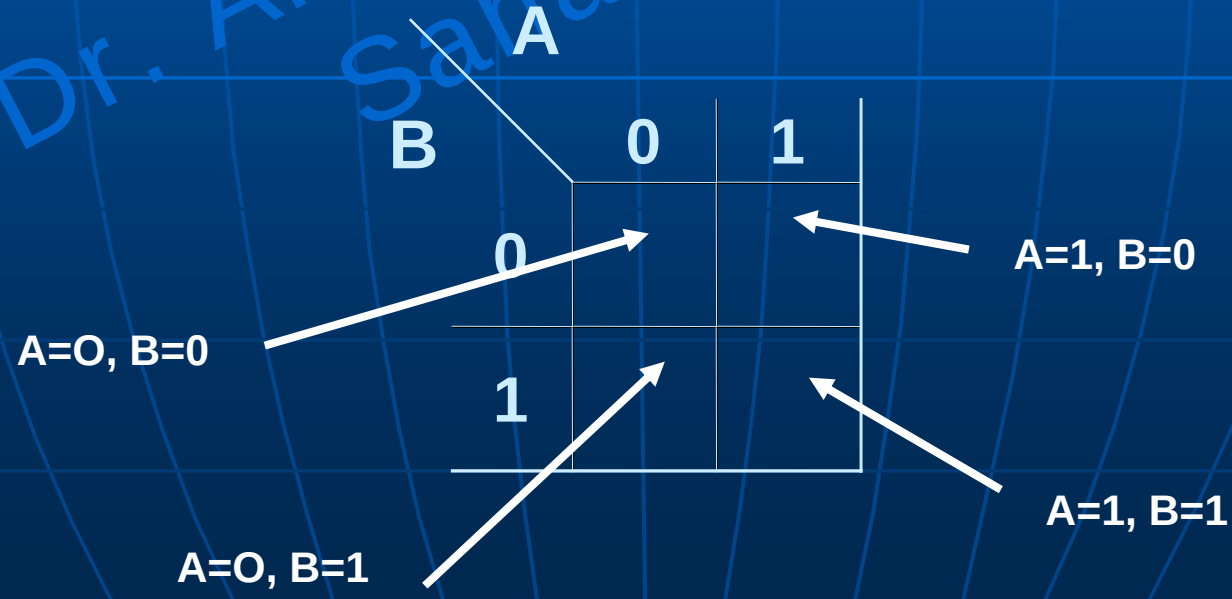
Map Simplification

- Karnaugh map: a diagram made up of squares, with each square representing one minterm of the function.

  -> be done systematically

  -> Simpler to find the minimum solution

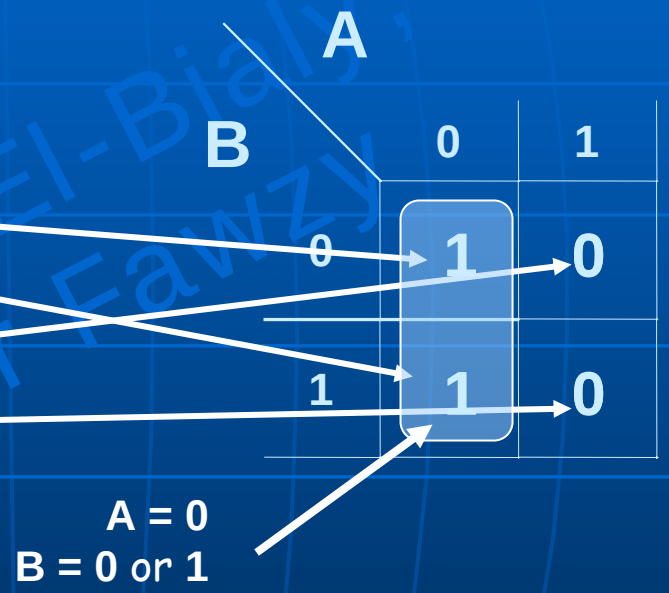
- Two variables K-map:



Map Simplification

- Example of a two-variable K-map:

A	B	F
0	0	1
0	1	1
1	0	0
1	1	0



F = A'B' + A'B

F = A'

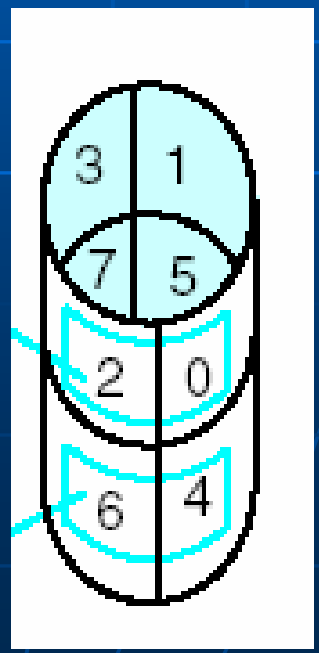
Map Simplification

- Three variables K-map:

  The order of BC variables are 00, 01, 11 and 10. They are Gray code (i.e. only one bit is changed each time).

- Each minterm corresponds to a location on K-map: $m_0 \ldots m_7$

A \ BC	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6



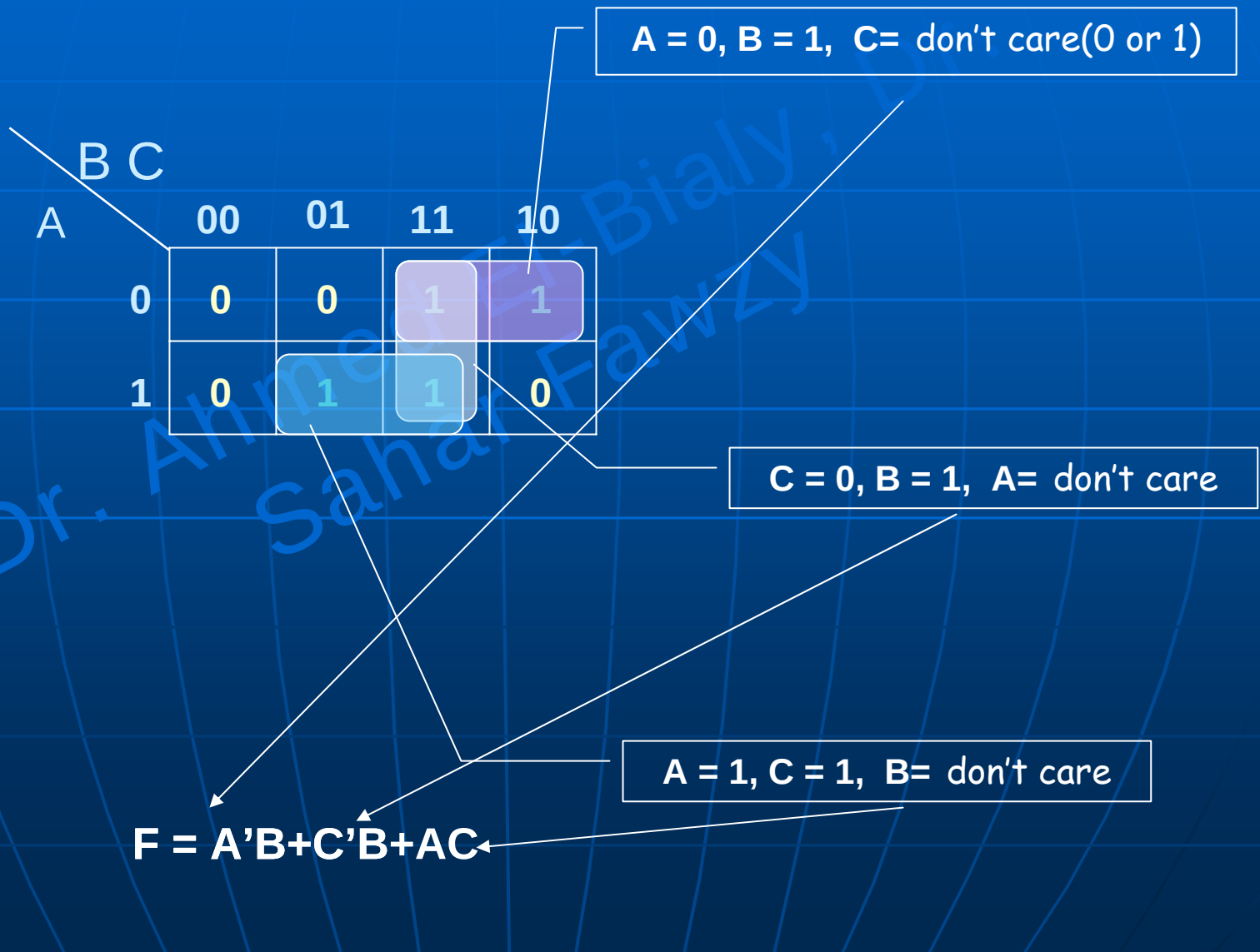
Map Simplification

- Example of a three-variable K-map:

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$$F(A,B,C) = \sum m(2, 3, 5, 7)$$

Map Simplification



Map Simplification

- Four variables K-map :

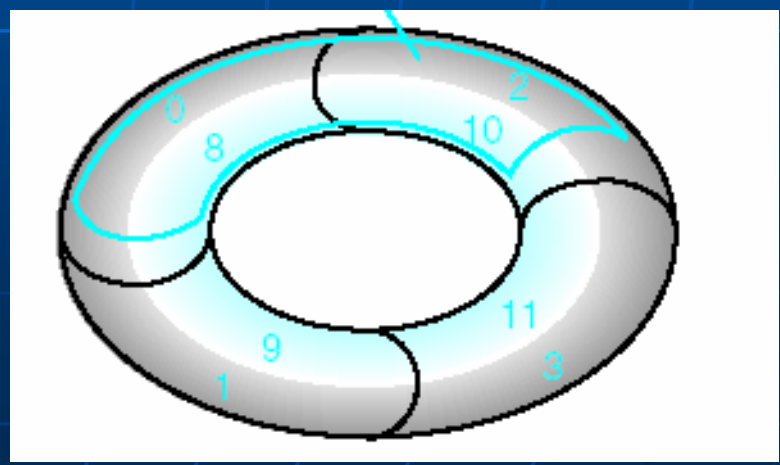
  - Note that the orders of AB and CD follow Gray code.

- Each minterm corresponds to a location on K-map: $m_0 \ldots m_{15}$

- :

.

AB \ CD	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}



Map Simplification

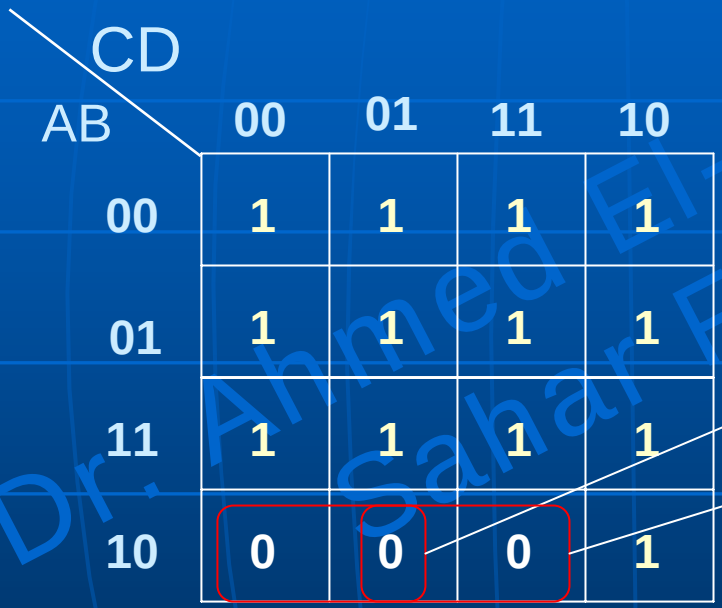
- Example of a four-variable K-map:

Simplify function F = Σ m (1, 2, 3, 5, 7, 10, 14, 15)

F =A’D+A’B’C+ABC+ { ACD’ or B’CD’

Map Simplification

- Sometimes it is easier to plot the product of sum:



F' =AB'C'+AB'D

F =(AB'C'+AB'D)'

= (A'+B+C)(A'+B+D')

Map Simplification

- K-Map with “Don’t Care” Values :

  - “Don’t care conditions” means unspecified minterms of a function, which is marked with “X”

F (A,B,C,D)
= Σ m (1, 5, 6, 10, 11, 14)
+ Σ d (0, 7, 9, 15)
= AC + BC + A’C’D

AB \ CD	00	01	11	10
00	X	1	0	0
01	0	1	X	1
11	0	0	X	1
10	0	X	1	1

Thank you

see you in

Sequential Circuits